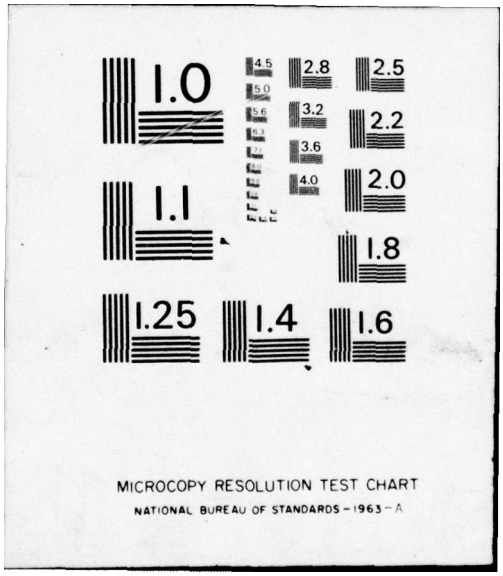
AD-A031 578 NAVAL RESEARCH LAB WASHINGTON D C F/G 9/2
ANOMALOUS BEHAVIOR OF THE FIFTY-PERCENT RULE.(U)
SEP 76 J E SHORE
UNCLASSIFIED NRL-MR-3368 NL

END

DATE FILMED
12-76


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

FG (12)

NRL Memorandum Report 3368

Anomalous Behavior of the Fifty-Percent Rule

JOHN E. SHORE

Information Systems Staff
Communications Science Division

September 1976

NAVAL RESEARCH LABORATORY
Washington, D.C.

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: NRL Memorandum Report 3368	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)*: ANOMALOUS BEHAVIOR OF THE FIFTY-PERCENT RULE	5. TYPE OF REPORT & PERIOD COVERED: Interim report on a continuing NRL problem.
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): John Shore / John E. Shore	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Research Laboratory, Washington, D.C. 20375	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NRL Problem B02-35, 61153N, RR014-09-41
11. CONTROLLING OFFICE NAME AND ADDRESS: Department of the Navy, Office of Naval Research, Arlington, Virginia 22217	12. REPORT DATE: September 1976
	13. NUMBER OF PAGES: 28
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*: UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

NRL-MR-3368

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Dynamic memory allocation
Fifty-percent rule
Storage fragmentation

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This paper reports simulation data showing that, in dynamic memory allocation, the average free-to-allocated-block ratio can be considerably less than predicted by the fifty-percent rule. A new derivation is given, and it is shown that previous derivations make an implicit assumption that may be violated frequently. Based on the simulation data and on the derivation, it is hypothesized that the anomalous behavior results from the combined effects of systematic placement and the statistics of the release process. Additional simulations supported this hypothesis. Systematic placement, which refers to the

(Continues)

20. Abstract (Continued)

natural convention of always allocating storage requests against the same end of the free block selected by the allocation strategy, tends to order blocks within contiguous groups according to their allocation time. The degree of anomalous behavior depends on the extent to which allocated blocks are released in the order of their allocation. For non-Markovian release processes the extent of the correlation between allocation order and release order varies inversely with the coefficient of variation of the memory residence time distribution. For values less than about .8, the free-to-allocated-block ratio can be considerably less than 1/2. For larger values, Knuth's estimate appears to be accurate provided that the average number of allocated blocks is high. Some practical implications are discussed briefly.

CONTENTS

ANOMALOUS BEHAVIOR OF THE FIFTY-PERCENT RULE

1.0 INTRODUCTION

Dynamic storage allocation strategies address the problem of choosing, from among a set of available blocks of storage, one in which to allocate a given storage request. If storage requests are rounded up to the nearest multiple of some allocation unit, then the storage wasted by this procedure is referred to as internal fragmentation [1]. Whether storage requests are rounded up or not, some available storage is usually wasted because of the proliferation of blocks of available storage that are too small to satisfy a pending request. This phenomenon is referred to as external fragmentation [2]. In general, the choice of allocation strategy will affect a variety of dynamic performance measures, such as the amount of fragmentation or the efficiency of using the total amount of storage.

It would be nice if one could study the relative merits of different allocation strategies by means of mathematical analysis, and thereby reach more general conclusions than those that tend to result from such empirical studies as [1-9]. Mathematical analysis is generally feasible in the case of internal fragmentation, because internal fragmentation depends only on the probability distribution of request sizes and on the round-up rules. In most cases, neither the request distribution nor the round-up rules depend on the current memory state, so that one can compute the expected internal fragmentation without reference to the dynamics of storage allocation and release. Examples are [10,11].

Although some results have been obtained [12,13], external fragmentation is generally less amenable to analytical study. The main difficulty seems to be

Note: Manuscript submitted August 31, 1976.

that, in attempting to write a mathematical definition of a given allocation strategy, one obtains a rather perverse function of the current memory state in a form that defies the closed-form computation of expected values for various dynamic functions of the memory state. A less ambitious approach is to attempt analysis that is valid for large classes of allocation strategies, a procedure that can obviate dealing with functions dependent on the memory state. The "fifty-percent rule" is a well known result of such an analysis. First derived by Knuth [3], the fifty-percent rule gives an estimate for the quantity ρ defined by

$\rho \equiv$ equilibrium ratio of the average number of free blocks of storage to the average number of allocated blocks.

An allocated block is a contiguous sequence of memory words in which is allocated a single storage request. A free block is a contiguous sequence of unallocated words delimited on both ends by allocated blocks or by one end of the memory. Knuth's analysis resulted in an approximate estimate ρ^* of ρ given by

$$\rho^* \approx \frac{1}{2} p_f \quad , \tag{1.1}$$

where

$p_f \equiv$ probability that any given allocation will not exactly fill the selected free block (i.e., the probability that a fragment remains).

In situations where request sizes are infrequently equal, the probabiliy p_f will be close to 1 and the ratio ρ therefore close to $\frac{1}{2}$. Other derivations have ignored the possibility of exactly-fitting allocations, with the result that the estimate $\rho \approx \frac{1}{2}$ is obtained directly [14].

In analyzing the simulation data reported in [7], I was surprised to find that the ratio ρ was significantly less than the estimate given by Eq.(1.1). These observations led to additional studies that are the subject of the present paper.

The next section describes the simulation that produced the anomalous observations, the observations themselves, and how the anomaly could be made to disappear by modifying the manner in which requests are placed in the free block selected by the allocation strategy. Section 3 contains derivations that illuminate the anomalous behavior. It is shown that previous derivations [3,14] contain an implicit assumption that may be violated frequently. Section 4 contains the results of new simulations that support the conclusions reached from the derivations in Section 3. General conclusions are discussed in Section 5.

2.0 THE ANOMALOUS BEHAVIOR

2.1 Simulation Description

The following description of the simulation is taken from [7], where additional details can be found.

The simulator attempts to allocate an infinite proffered workload of storage requests in a fixed-size memory. Storage is allocated in units of one word so that no more than the requested amount is ever allocated. Resulting fragmentation is therefore purely external. The event-driven simulator maintains a list of available-storage blocks. When an allocated block is released, the simulator determines whether it is located adjacent to one or two blocks already on the available-storage list. If so, the appropriate blocks are combined by suitable modifying the available-storage list. If not, the newly released block is added to the list. The simulator then attempts to allocate the pending storage request. If there is space for it, the request is allocated as the low-address end of whichever available block is selected by a prespecified allocation strategy. The available-storage list is modified, and the newly-allocated space is scheduled for release at a time randomly selected from a prespecified distribution. Following a successful allocation, the simulator continues

to generate and allocate new requests until an attempted allocation fails. Whenever failure occurs, the simulator computes various quantities of interest, such as the number of free blocks, their average size, etc., and updates summary statistics. The simulator then advances to the next-scheduled storage release and continues as described in the foregoing until a prespecified time limit is reached. Summary statistics are then stored and the simulator begins anew after clearing memory and reinitializing everything but the random number generator. During one run, the simulation is repeated for a prespecified number of such iterations, N, after which the statistics for each iteration and averages over the ensemble of iterations are printed.

Among the data available after each run are the average number of allocated and free blocks for each of the N iterations, as well as the variances of these quantities. These data are sufficient to compute the ratio ρ for each iteration and then the average of ρ over the ensemble of iterations. This average provides the best estimate of ρ for the simulation conditions. From the variance in the ratio for each iteration, and from the variances in the quantities that make up these ratios, one can judge the accuracy of the overall estimate of ρ.

One cannot ignore the question of whether or not the averages computed by the simulator correspond to time averages. In general, the answer depends both on the quantity being sampled and on the sampling process. For an example of simulation averages that do not correspond to time averages, see [7]. As mentioned earlier, samples of the number of allocated and free blocks are taken after each allocation failure. Since allocation is attempted each time storage is released, and since allocation takes no time, samples are therefore taken at the time of every storage release. In all of the simulations reported herein, newly allocated blocks were scheduled for release so that their residence time

was uniformly distributed between prespecified limits. Therefore, when the average numbers of allocated and free blocks are computed from sufficiently numerous samples, each taken after an allocation failure, the resulting quantities are valid estimates of time averages.

All of the quantitative results reported in this paper are based on simulations of a 32K word memory using first-fit allocation and exponentially distributed request sizes. Results obtained using best-fit allocation were quite similar. The general nature of the results do not seem to be sensitive to the shape of the request size distribution, provided that request sizes are infrequently equal to each other. I did not, however, perform systematic studies with non-exponential distributions of request sizes.

2.2 Initial Simulation Results

In Fig. 1, the points marked with circles (●) show the results of a series of experiments in which the ratio ρ was measured as the mean request size $\bar{r}$ was varied. The computation of these points can be clarified best by using some new notation. Each iteration i of the simulation produces the following numbers:

$E_i(A)$ ≡ average number of allocated blocks

$\sigma_i(A)$ ≡ variance of the number of allocated blocks

$E_i(F)$ ≡ average number of free blocks

$\sigma_i(F)$ ≡ variance of the number of free blocks

In all of the experiments recorded in Fig. 1, the memory residence times of requests were uniformly distributed between 5.0 and 15.0 time units. Typically, about 6000 samples contributed to each $E_i(A)$ and $E_i(F)$. These samples were taken after the transients had died down. The coefficients of variation $\sigma_i(A)/E_i(A)$ and $\sigma_i(F)/E_i(F)$ were generally less than .3, which indicates that the averages $E_i(A)$ and $E_i(F)$

are meaningful measures. The free-to-allocated-block ratio for each iteration was computed as $\rho_i = E_i(A)/E_i(F)$. Each point ● plotted in Fig. 1 is the average of ρ_i over ten iterations of the sample. Given the ten samples available in each case, these points are the best estimates of the mean of the distributions of ρ_i. The error bars show the 95% confidence limits of these estimates.

Also plotted in Fig. 1 (using triangles (▲)) are the estimates ρ^* predicted by Eq. 1.1. Values for p_f were estimated from simulation data giving, for each iteration, the number of exactly-fitting allocations and the total number of allocations. Each point is the result of averaging data from 10 iterations. The 95% confidence limits of these mean estimates are about the size of the symbols ▲ used to mark the points.

The results in Fig. 1 show the measured values of the free-to-allocated-block ratio to be significantly less than those predicted by Eq. 1.1. At the time when I first noticed the anomaly, a colleague hypothesized that it might be related to the simulation convention of always allocating storage requests against the same end of the free block selected by the allocation strategy. That is, when the size of a selected free block exceeds that of the current storage request, one has three basic choices of where in the selected free block to place the current request——namely, at either end of somewhere in between. Placing it in between seemed foolish since that would generate an extra fragment. Given that the current request was to be placed at one end of the selected block, it seemed natural to select one end (either the "low address" end or the "high address" end) and systematically allocate against that end every time. This convention of "systematic placement," as I shall call it, was used in the simulations reported in [7] and corresponds to Knuth's "Algorithm A" [3].

Following up on the hypothesis of my colleague, I modified the simulation so that each request was allocated against a pseudo-randomly chosen end of the selected free block. The two choices were made equiprobable. I shall refer to this alternative convention as "random placement." The points in Fig. 1 marked with the symbol X show the results of repeating the measurements of ρ with random placement instead of systematic placement. These results are reasonably close to the predicted values ρ^*, which themselves were not appreciably affected by the change to random placement.

In general, the use of systematic placement resulted in both fewer free blocks and more allocated blocks compared to random allocation. This better use of storage was reflected in the time-memory product efficiency. If n requests of r_i words $i = 1,2,\ldots,n$, *are allocated for times* t_i *in a memory of size* M during a total elapsed time T, then the efficiency E is

$$E = \frac{1}{MT} \sum_{i=1}^{n} r_i t_i \quad .$$

E is a direct measure of how well the memory has been used. The general question of performance measures for dynamic allocation is discussed in [7]. The efficiencies obtained in the systematic and random placement experiments are shown in Fig. 2. The data for each point in Fig. 2 were taken from the same run as the corresponding point in Fig. 1.

3. THE BEHAVIOR EXPLAINED

3.1 Derivations

Some insight into the results of the previous section can be gained by considering Denning's derivation of the fifty percent rule [14]. His derivation rested on the assumption that, during the time an allocated block spends in memory, half of the transactions in the region immediately above the block are allocations and half are releases. It is easy to see that this assumption might

be better satisfied with random placement than with systematic placement, since systematic placement introduces an obvious asymmetry into the allocation process. This is not, however, the whole story. There is a less obvious, but equally important asymmetry in the release process. This release asymmetry can be seen with the help of the following derivation, which is motivated by that of Knuth [3].

Allocated blocks can be classified according to whether or not the spaces immediately above them and below them are also allocated. There are three basic types (see Fig. 3):

a - There are free blocks on both sides. Release of a type a block decreases by one the number of free blocks.

b - There is a free block on one side. Release of a type b block has no effect on the number of free blocks.

c - There are allocated blocks on both sides. Release of a type c block increases by one the number of free blocks.

We shall need the following notation:

N_a, N_b, N_c = average number of allocated blocks of type a, type b, and type c, respectively.

$N = N_a + N_b + N_c$ = total average number of allocated blocks

R_a, R_b, R_c = average total release rates for type a block, type b blocks, and type c blocks, respectively.

r_a, r_b, r_c = average release rates per unit block for type a blocks, type b blocks, and type c blocks, respectively.

F = average number of free blocks

A = average total allocation rate

A_e = average total rate of exactly fitting allocations (each such allocation decreases by one the number of free blocks)

The average numbers of free and allocated blocks are related by the equation

$$F = \frac{1}{2}\,(2N_a + N_b + \bar{\varepsilon})\,, \tag{3.1}$$

where $\bar{\varepsilon}$ is the average of a quantity ε that is determined by the boundary conditions as follows:

$$\varepsilon = \begin{cases} 0\,, & \text{if both the first and the last words in memory are allocated} \\ 1\,, & \text{if either the first or the last words in memory are allocated, but not both} \\ 2\,, & \text{if neither the first nor the last words in memory are allocated.} \end{cases} \tag{3.2}$$

By the introduction of $N = N_a + N_b + N_c$, Eq.(3.1) can be transformed into

$$\rho = \frac{F}{N} = \frac{1}{2}\left(1 + \frac{\bar{\varepsilon}}{N}\right) - \frac{N_c - N_a}{2N}\,. \tag{3.3}$$

Since the average number of free blocks is constant once equilibrium is reached, for each decrease in the number of free blocks there must be a corresponding increase. Hence, the following balance equation holds:

$$A_e + R_a = R_c \quad . \tag{3.4}$$

Rewriting Eq.(3.4)as $A_e + N_a r_a = N_c r_c$, solving for $N_c - N_a$, and substituting the result into Eq.(3.3) yields

$$\rho = \frac{1}{2}\left(1 + \frac{\bar{\epsilon}}{N}\right) - \frac{A_e}{2Nr_c} - \frac{N_a}{2N}\left(\frac{r_a}{r_c} - 1\right) . \tag{3.5}$$

Thus, there are two effects that can increase the difference in the average number of type c and type a blocks $(N_c - N_a)$. One is determined by the rate of exactly fitting allocations. The other is determined by the relative per-unit-block release rates of type a and type c blocks.

Suppose that

$$r_a = r_b = r_c \tag{3.6}$$

holds, as was assumed implicitly by Knuth [3]. This assumption means that, at the time of each storage release, any given type a block is as likely to be released as any given types b block or type c block. The last term in Eq. (3.5) is wiped out, leaving

$$\rho = \frac{1}{2}\left(1 + \frac{\bar{\epsilon}}{N}\right) - \frac{A(1 - p_f)}{2Nr_c} \quad , \tag{3.7}$$

where we have introduced the relation $A_e = A(1 - p_f)$. Since we have assumed Eq. (3.6), the overall balance equation $A = R_a + R_b + R_c$ reduces to

$$\begin{aligned} A &= r_c(N_a + N_b + N_c) \\ &= Nr_c \quad . \end{aligned}$$

Eq. 3.7 then becomes

$$\rho = \frac{1}{2}\left(p_f + \frac{\bar{\epsilon}}{N}\right)$$

$$= \rho^* + \frac{\bar{\epsilon}}{2N} \quad , \tag{3.8}$$

where ρ^* is Eq.(1.1), Knuth's estimate of ρ. For both systematic and random placement, it is unlikely that $\bar{\epsilon}$ varies much from $\epsilon \approx 1$ (see Eq. 3.2) so that

$$\rho \approx \rho^* + \frac{1}{2N}$$

$$\approx \rho^* \quad , \tag{3.9}$$

provided $N \gg 1$. In the systematic placement runs plotted in Figs. 1-2, the values of N for $\bar{r}$ = 256, 512, 1024, and 2048 were N = 106.80, 52.57, 25.89, and 12.98, respectively.

Now consider the validity of Eq.(3.6). Suppose that the dynamics of storage allocation and release result in type a blocks being fewer and relatively older than type b or c blocks. Type a blocks would then be relatively likely, per-unit-block, to be released at any given time. In this case, Eq.(3.6)would be violated by $r_a > r_c$, and the last term in Eq.(3.5)would produce a decrease in the ratio ρ. Indeed, in the simulations discussed in Section 2, the combination of systematic placement with uniformly distributed residence times has the foregoing effect. In order to see why, one must realize first that systematic placement results, on the average, in each block within a contiguous group of allocated blocks being older than the block immediately below it, and the bottom block of such a group being younger than the top block of the group below.

Consider a group of allocated blocks such as shown in Fig. 4. Suppose that the top block $\underline{b}_1$ was allocated at time t_1 and the next block $\underline{c}_2$ was allocated at time t_2. It is possible for $\underline{b}_1$ to be younger than $\underline{c}_2$ ($t_1 > t_2$), but this would only be true if $\underline{b}_1$ resulted from an exactly fitting allocation, which is quite unlikely. More likely, but still relatively unlikely, is the possibility that the two blocks were allocated simultaneously in a large free block ($t_1 = t_2$). Most likely is the case of $\underline{c}_2$ being allocated sometime after $\underline{b}_1$ after sufficient space had accumulated below $\underline{b}_1$ ($t_1 < t_2$). The foregoing argument applies to any pair of adjacent blocks, so that $t_1 < t_2 < t_3 < t_4$ holds on the average. Now, if residence times are uniformly distributed in the interval $[t_a, t_b]$, the block $\underline{b}_1$ will be released sometime between $t_1 + t_a$ and $t_1 + t_b$. The block $\underline{c}_2$ will be released sometime between $t_2 + t_a$ and $t_2 + t_b$. Since $t_1 < t_2$, the block $\underline{b}_1$ will be released on the average prior to $\underline{c}_2$. Whenever

$$(t_2 - t_1) \geq (t_b - t_a) \tag{3.10}$$

holds, $\underline{b}_1$ will always be released prior to $\underline{c}_2$. Similarly, each block in a contiguous group is likely to be released prior to the block immediately below it. Since it is unlikely that the free space below $\underline{b}_4$ resulted from the release of a block allocated after $\underline{b}_4$, the top block in the group beginning below the free space tends to be older than $\underline{b}_4$ and therefore likely to be released prior to $\underline{b}_4$. If it is released prior to $\underline{b}_4$, then sufficient free space may be available for a new allocation against the bottom of $\underline{b}_4$. This tendency for young blocks to join the bottom of a contiguous group accompanies the tendency for the oldest block to be released from the top of the group. Thus, each contiguous group of blocks tends to migrate downward in memory.

Whenever Eq. (3.10) does not hold, it may happen that $\underline{c}_2$ is released before $\underline{b}_1$, even though $\underline{b}_1$ is older. The frequency of such an event increases with the magnitude of $(t_1 + t_b) - (t_2 + t_a)$. If the resulting type $\underline{a}$ block is released before any allocation is made underneath it, its release will contribute to r_a. This will happen a significant fraction of the time since the space released by $\underline{c}_2$ is only of average size, and since the former $\underline{b}_1$ is likely to be released before even more space is made available by the release of $\underline{c}_3$. Since the former $\underline{b}_1$ is close to the end of its life, it is relatively likely to be released at any given time; thus, the per-unit-block release rate of such blocks is higher than r_b or r_c. The result will be $r_a > r_c$, provided that the foregoing mechanism makes the principal contribution to N_a. It is not difficult to see that systematic allocation makes infrequent the production of type $\underline{a}$ blocks by other mechanisms. One example is the transformation of $\underline{b}_4$ into a type $\underline{a}$ block. Although $\underline{c}_3$ is likely to be released before $\underline{b}_4$, in most cases additional allocations already will have been made underneath $\underline{b}_4$. Even if $\underline{b}_4$ does become of type $\underline{a}$, it is unlikely to remain so for long.

We may conclude that systematic placement indeed results in $r_a > r_c$, which in turn produces a significant decrease in the free-to-allocated-block ratio ρ (Eq. 3.5). It should be obvious that the mechanisms that strongly favor $r_a > r_c$ for systematic placement do not exist for random placement. Thus, the anomalous behavior of ρ in Fig. 1 is explained. Since ρ appears to be slightly less than ρ^* in the random placement case, the data suggest that r_a is slightly greater than r_c for random placement as well, albeit much less so than for systematic placement. This might be explained by noting that, since no newly allocated block is of type $\underline{a}$, type $\underline{a}$ blocks tend to be older than average and therefore more likely to be released.

It is important to realize that the explanation for the decreased value of ρ in the case of systematic placement depended on more than the concept of systematic

placement. It also depended on the release process——namely, that the release time for each block was determined at allocation time according to a uniform distribution of residence times. Thus, the longer a block was in memory, the more likely it was to be released. The explanation of $r_a > r_c$ depended on this correlation; without it, the explanation would be invalid. For example, suppose that a given block is released, *not when a prespecified time is reached*, but when the block is selected at random from the set of allocated blocks. In this case it is obvious that $r_a = r_b = r_c$ holds and that ρ will be given by Eq. (3.8) whether or not systematic allocation is practiced.

3.2 Additional Simulation Results

In Section 3.1, based on simulation data (Section 2.2) and the derivation of Eq. (3.5), I developed the hypothesis that the anomalous behavior results from the combined effects of systematic placement and the statistics of the release process. This development included an explanation for r_a being significantly larger than r_c in the case of systematic placement. Because it was verbal, qualitative, and filled with repeated applications of such weak relations as "on the average more likely than," this explanation leaves much to be desired. The main impediment to better, analytic explanations is the non-Markovian nature of the simulation dynamics. Fortunately, the verbal explanation suggests additional simulation experiments and predicts their results. Thus, the hypothesis can be tested.

For any two adjacent allocated blocks, if t_1 is the allocation time of the upper block and t_2 is the allocation time of the lower block, the upper block will always be released first if Eq.(3.10) holds. If Eq.(3.10) does not hold, the probability that the upper block will be released first increases as the quantity $(t_1 + t_b) - (t_2 + t_a)$ decreases. As described in Section 3.1, the effect producing $r_a > r_c$ with systematic placement depends on the relatively high probability of the upper block being released first. The higher this probability, the more significant will be the third term in Eq. (3.5), and the smaller will be the free-to-allocated-block ratio ρ. The foregoing suggests a series of experiments in

which ρ is measured as the range of memory residence times $(t_b - t_a)$ is varied. For large values of $(t_b - t_a)$, the effect of systematic placement on r_a should begin to disappear, and ρ should approach the random placement and ρ^* values.

Fig. 5 shows the results of such an experiment. The mean request size for all runs was 1024. The mean request residence time $\bar{t} = (t_b + t_a)/2$ was held constant at $\bar{t} = 100$ while the width of the uniform distribution $[t_a, t_b]$ was varied. The distributions used were [90,110], [75,125], [50,150], [25,175], and [0,200]. This variation is plotted on the abcissa using the coefficient of variation (standard deviation divided by mean) of the residence time distribution $\alpha = (t_b - t_a)/(\bar{t}\sqrt{12}) = (t_b - t_a)/346.41$. As predicted, the free-to-allocated-block ratio ρ for systematic placement is affected strongly by α. For random placement, ρ is close to ρ^* in all cases. The significance of the small apparent slope in the data for random placement is uncertain. Not shown in Fig. 5 is the result of an additional experiment in which an exponential distribution of residence times was used. The coefficient of variation for the exponential distribution is $\alpha = 1$. The mean residence time was $\bar{t} = 100$ and all other variables were the same as those for the runs shown in Fig. 5. For systematic placement the average value of ρ was $\rho = .502$. For random placement the average value was $\rho = .497$ Thus, for the widespread residence times produced by an exponential distribution, the beneficial effect of systematic placement disappears.

As in the simulations discussed in Section 2.2, systematic placement resulted in higher efficiency E than did random placement (see Fig. 6). The data for each point in Fig. 6 were taken from the same run as the corresponding point in Fig. 5. For the experiment with exponentially distributed residence times (not shown), the average efficiency for systematic placement was E = .757 and the average efficiency for random placement E = .755.

4.0 DISCUSSION

The results in Section 3 show that the anomalous behavior of the fifty-percent rule is explained well by the hypothesized combined effects of systematic placement and the statistics of the release process. Given systematic placement, which tends to order blocks within groups according to their allocation time, the extent of the anomaly depends on the extent to which allocated blocks are released in the order of their allocation. In the limit of perfect first-in-first-out behavior, there would never be more than two free blocks and allocated blocks of type a would never be produced. For non-Markovian release processes, the extent of the correlation between allocation order and release order is indicated by the coefficient of variation α of the memory residence time distribution. For values of α less than about $\alpha \approx .8$, the free-to-allocated-block ratio can be considerably less than 1/2, which means that the free space list is shorter and the performance better than implied by the fifty-percent rule. For larger values of α Knuth's estimate c^* appears to be accurate provided that the average number of allocated blocks is high (see Eq. (3.9)).

The results have implications for situations involving "priority interrupts" that require the immediate release of space. Meeting such a requirement by randomly selecting blocks for release would tend to equalize r_a, r_b, and r_c, and would therefore be bad. Better would be a policy that restricts the selection to type a blocks or, if there happen to be none, to type b blocks at the upper end of groups.

Memory compaction, a process that rearranges memory contents so that all free space is contiguous, is an expensive, brute force way of making the best use of available space. The results of Section 3 suggest that a modified compaction algorithm, in which type a blocks are moved down until they become type b, may prove to be advantageous in some situations.

Above all, like those in [7], the results reinforced my opinion that the dynamics of memory usage comprise complicated phenomena in which observable effects often have subtle causes. These phenomena, which are usually intuitive only after the fact, are best studied by the method of strong inference [15].

5.0 ACKNOWLEDGMENTS

I am grateful to H. Elovitz, for suggesting that systematic placement might explain the anomalous behavior, and to W. S. Ament and R. W. Johnson, for numerous helpful discussions.

Note: In all of the figures showing simulation results, the memory size was 32K, first-fit allocation was used, request sizes were exponentially distributed, and residence times were uniformly distributed. Points for systematic placement are plotted as ●. Points for random placement are plotted as X. Points showing the estimate ρ^* are plotted as ▲. Each point is the result of averaging ten iterations. The error bars show the 95% confidence limits of the estimates of the means.

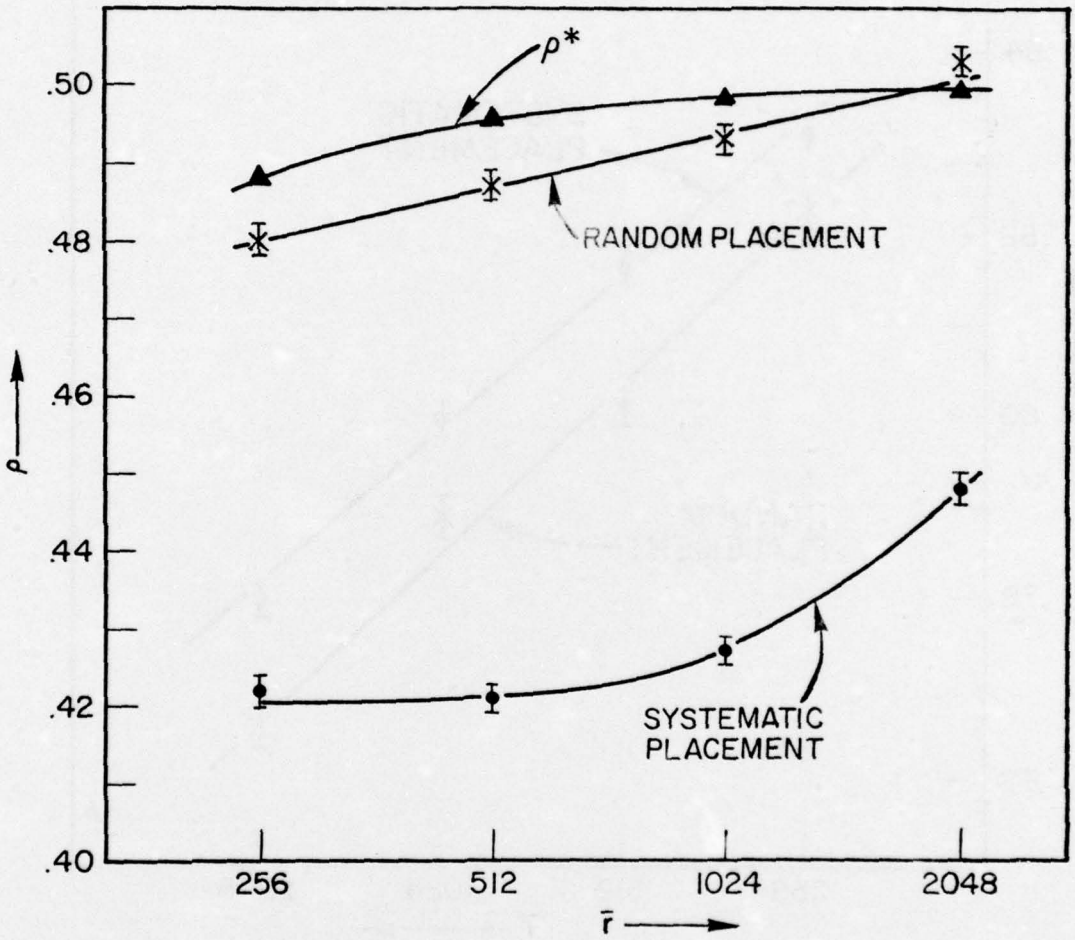


Fig. 1 — Free-to-allocated-block ratio ρ versus mean request size $\bar{r}$. Memory residence times were uniformly distributed in the interval [5,15].

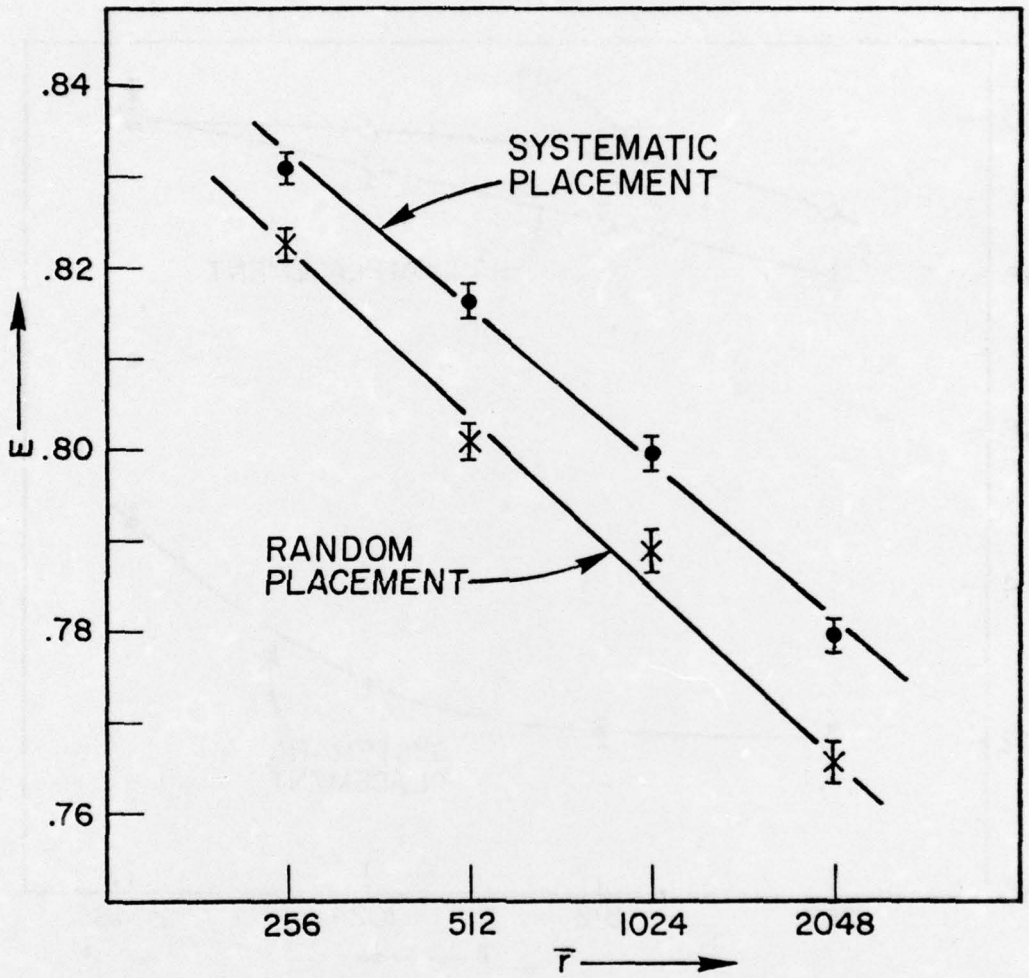


Fig. 2 — Allocation efficiency E versus mean request size $\bar{r}$. The data for each point were taken from the same run as the corresponding point in Fig. 1.

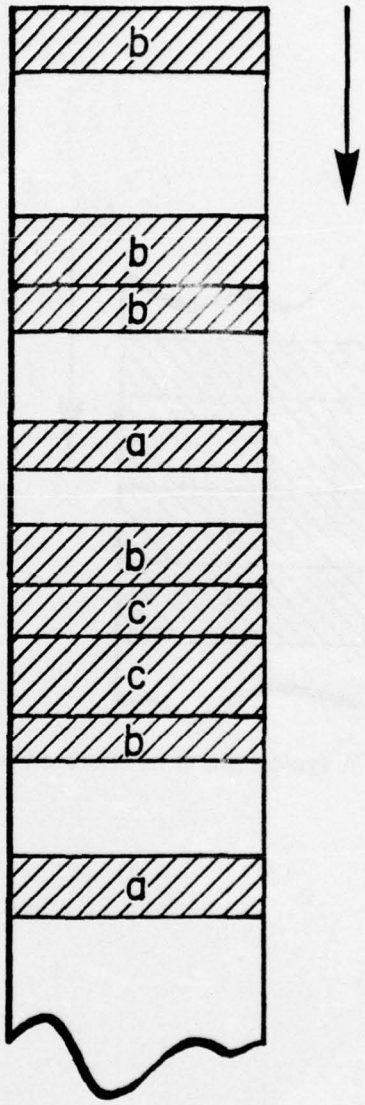


Fig. 3 — Diagram showing the three types of allocated blocks discussed in the text. Memory length is depicted by the vertical dimension of the diagram. The arrow points in the direction of increasing address. Cross-hatched areas represent allocated blocks and empty areas represent free blocks. In the simulations, systematic placement corresponds to allocating a new request against the upper end of a free block in this diagram.

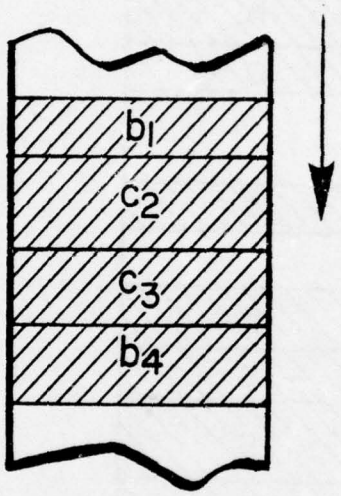


Fig. 4 — A typical group of allocated blocks

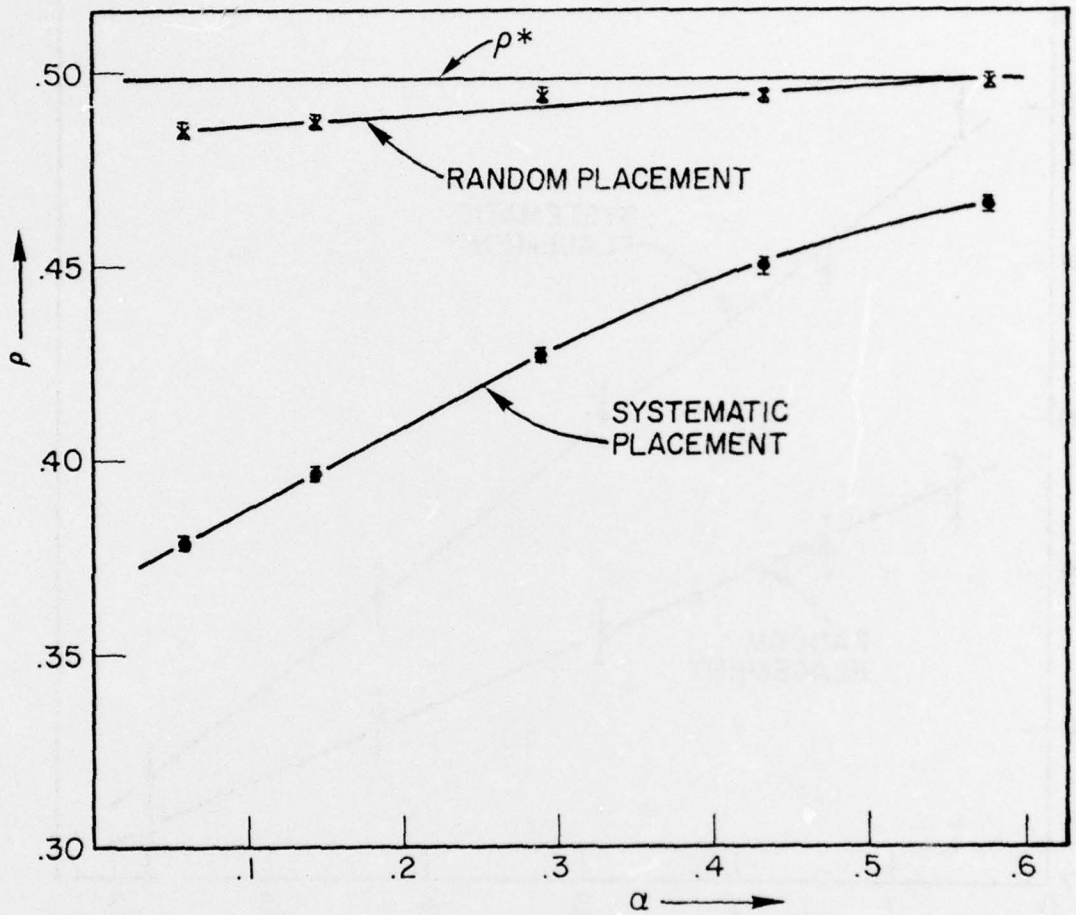


Fig. 5 — Free-to-allocated-block ratio ρ vs. coefficient of variation α of the memory residence time distribution. Residence times were uniformly distributed around a mean of 100. The mean request size was 1024. The line marked ρ^* shows the estimate ρ^*, which was not affected by α.

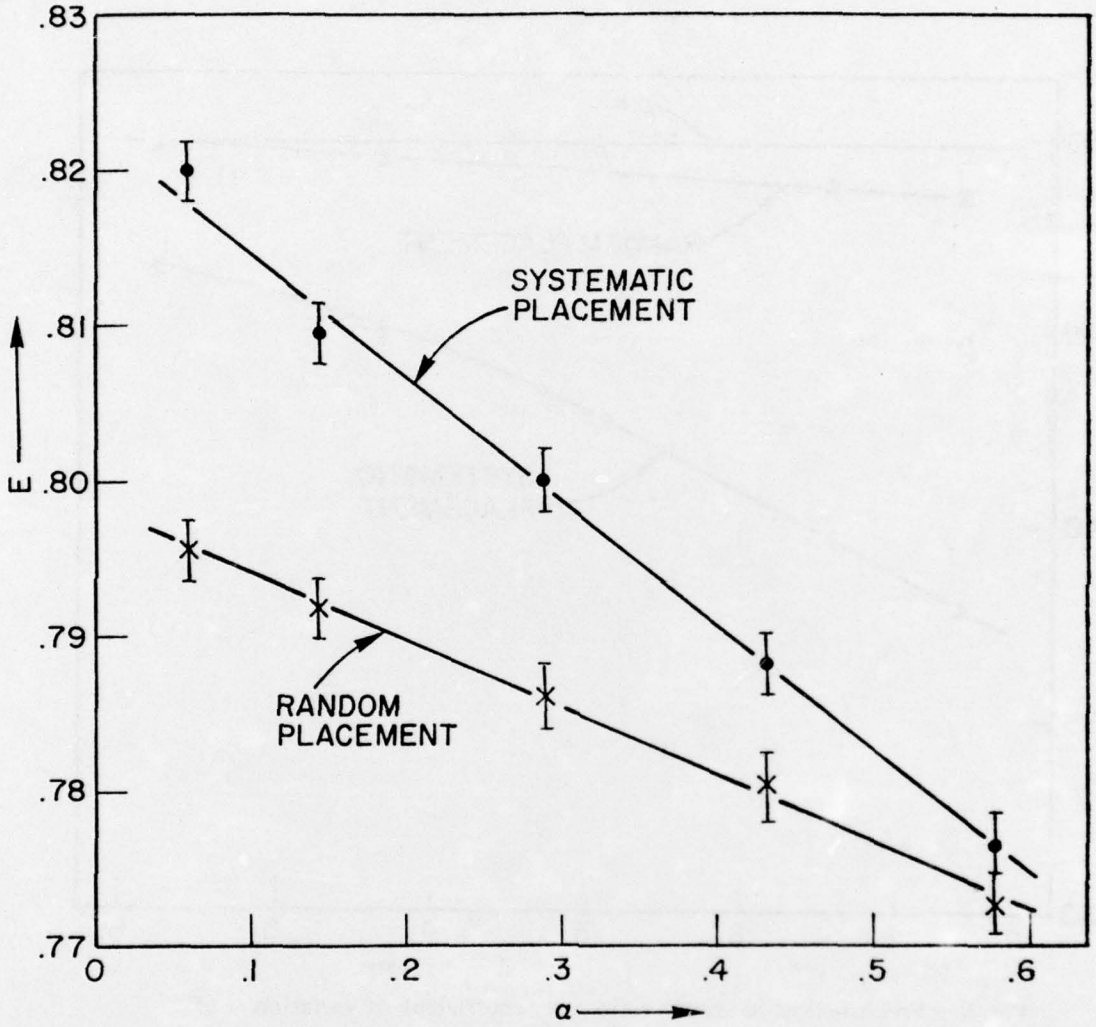


Fig. 6 — Allocation efficiency E versus coefficient of variation α of the memory residence time distribution. The data for each point were taken from the same run as the corresponding point in Fig. 5.

References

1. Randell, B., A note on storage fragmentation and program segmentation, Comm. ACM 12, 7 (July 1969), 365-372.

2. Collins, G. O., Experience in automatic storage allocation, Comm. ACM 4, 10 (Oct 1961), 436-440.

3. Knuth, D. E., The Art of Computer Programming, Vol. I Fundamental Algorithms, Addison-Wesley, Reading, Mass., 1968, Section 2.5.

4. Margolin, B. H., Parmalee, R. P., and Schatzoff, M., Analysis of free-storage algorithms, IBM SYST J. 4, (1971), 283-304.

5. Hirschberg, D. S., A class of dynamic memory allocation algorithms, Comm. ACM 16, 10 (Oct 1973), 615-618.

6. Shen, K. K., and Peterson, J. L., A weighted buddy method for dynamic storage allocation, Comm. ACM 17, 10 (Oct 1974), 558-562, and Comm. ACM 18, 4 (Apr 1975),202.

7. Shore, J. E., On the external storage fragmentation produced by first-fit and best-fit allocation strategies, Comm. ACM 18, 8 (Aug 1975), 433-440.

8. Fenton, J. S., and Payne, D. W., Dynamic storage allocation of arbitrary sized segments, Proc. IFIP 74, North-Holland Pub. Co., Amsterdam, 1974, 344-348.

9. Weinstock, C. B., Dynamic storage allocation techniques, Ph.D. thesis, Carnegie Mellon University, April 1976.

10. Wolman, E., A fixed optimum cell-size for records of varying lengths, J. ACM 12, 1 (Jan 1965), 53-70.

11. Gelenbe, E., Boekhorst, J. C. A., and Kessels, J. L. W., Minimizing wasted space in partitioned segmentation, Comm. ACM 16, 6 (June 1973), 343-349.

12. Purdom, P. W., and Stigler, S. M., Statistical properties of the buddy system, J. ACM 17, 4 (Oct 1970), 683-697.

13. Betteridge, T., An analytic storage allocation model, Acta Informatica 3 (1974), 101-122.

14. Denning, P. J., Virtual memory, Computing Surveys 2, 3 (Sep 1970), 153-189.

15. Platt, J. R., Strong inference, Science 146, (1964), 347-353.